IN THE SPECIFICATION:

Please amend the specification as follows:

[0001] This application is a continuation-in-part of assignee's pending application U.S. Serial No. 09/397,331, filed on September 14, 1999, entitled "Method and System for Copyright Protection of Digital Images Transmitted over ~~Networks."~~Networks," now U.S. Patent No. 6,298,446.

[0002] To supplement the off-screen protection, the present invention preferably incorporates the invention described in assignee's pending application U.S. Serial No. 09/397,331, filed on September 14, 1999, entitled "Method and System for Copyright Protection of Digital Images Transmitted over ~~Networks."~~Networks," now U.S. Patent No. 6,298,446. The invention described in U.S. Serial No. 09/397,331 protects data while it is on-screen. Thus, the present invention, when combined with the invention described in U.S. Serial No. 09/397,331 protects designated content both while it is on-screen and while it is off-screen.

[0032] FIG. 1A is an illustration of an HTML page with protected text being displayed by a web browser ~~with~~without the intervention of a decoder;

[0033] FIG. 1B is an illustration of an HTML page with protected text being viewed ~~without~~with the intervention of a decoder;

[0034] FIG. 1C is an illustration of a display of a source listing for the HTML page of ~~FIG. 1A~~FIG. 1B;

[0043] To supplement the off-screen protection, the present invention preferably incorporates the invention described in assignee's pending application U.S. Serial No. 09/397,331, filed on September 14, 1999, entitled "Method and System for Copyright Protection of Digital Images Transmitted over ~~Networks,"~~Networks," now U.S. Patent No. 6,298,446. the contents of which are hereby incorporated by reference. The invention described in U.S. Serial No.

09/397,331 protects data while it is on-screen. Thus, the present invention, when combined with the invention described in U.S. Serial No. 09/397,331 protects designated content both while it is on-screen and while it is off-screen.

[0046] Reference is now made to FIG. 1C, which is an illustration of a display of a source listing for HTML ~~page120~~page 120 of FIG. 1B. Such a display can be obtained by a "View Page Source" command within a web browser. Since HTML page 120 contains encrypted text, when a user views the source for HTML page 120 it reveals only encrypted text 160 -- even though the display of the page shows decrypted text.

[0049] Client computer 220 includes a receiver 230 that receives the page and transfers it to a formatter 240 for determining a page layout, as described hereinbelow. After formatter 240 determines a page layout, a renderer 250 is responsible for ~~renders~~rendering the page into a graphics device ~~260~~, as described hereinbelow. By way of example, renderer 250 may be a web browser, which renders HTML pages. Also, by way of example, graphics device 260 may be a memory device, a screen device or a graphics port. Within the Microsoft Windows operating system, Netscape Communicator renders HTML pages directly into a screen device, and Microsoft Internet Explorer renders HTML pages into a memory device. Within the Macintosh operating system, both Netscape Communicator and Microsoft Internet Explorer render HTML pages into a graphics port.

[0051] The operation of formatter 240 will now be described. Formatter 240 determines a page layout for a given page. Typically, formatter 240 determines how many words to place within lines of the given page, based on the font type and font size currently selected. To determine widths of words, formatter 240 sends character strings to a string size module 280. String size module 280 accepts a character string as input, and returns the width of the string, based on the font type and font size currently selected. Formatter 240 repeatedly sends individual words to string size module 280, or strings with multiple words therein, in order to identify widths of text and thereby determine how many words to fit within lines of the page. Formatter 240 passes a page layout to renderer 250. String size module 280 is typically an operating system function, such as the Microsoft Windows GetTextExtent() function.

[0052] The operation of renderer 250 will now be described. Renderer 250 sends content such as text to a content output module 290. Content output module 290 accepts content as input and converts the content to graphics output, such as raster output or vector output, for writing to graphics device 260. Content output module 290 is typically one or more operating system functions, such as the Microsoft Windows TextOut() function and the Macintosh DrawText() function.

[0081] Reference is now made to FIG. 6, which is a simplified block diagram of a system for protection of content within a page including a formatting module, according to a preferred embodiment of the present invention. FIG. 6 includes the elements of ~~Fig. 4~~FIG. 4, and additionally includes formatter 240, decoder 610 and string size module 280. Formatter 240 calls string size module 280 to identify widths of various character strings, relative to the font types and font sizes of a device context, in order to determine a page layout. Specifically, formatter 240 uses character string width information to determine how many words to fit in lines of the page. Decoder 610 intercepts the character strings on their way to string size module 280, and replaces them with decrypted strings prior to string size module 280 determining the string widths. The intervention of decoder 610 ensures that the string widths provided to formatter 440 for determining a page layout correspond to string widths for decrypted strings, rather than for encrypted strings which typically have different word widths.